Command and Authorization Services for Multiple Agents Acting on an Advanced Life Support System

Cheryl Martin, Debra Schreckenghost, and Pete Bonasso

NASA Johnson Space Center
TRACLabs
1012 Hercules, Houston, TX, 77058, USA
cmartin@traclabs.com, ghost@ieee.org,
r.p.bonasso@jsc.nasa.gov

Abstract. This paper describes current work developing command and authorization services to support the coordination of multiple humans and an autonomous control agent working on the same underlying advanced life support system. The primary goal of these services is to prevent unknowing or accidental conflicts from arising as a result of issuing commands or taking action on the system. Avoiding such conflicts minimizes the risk of interfering with the work of another agent or putting the system into an unsafe operating state. This paper provides an overview of the advanced life support system at NASA to which this work has been applied and then discusses details for authorization, overrides, and system reconfiguration for commanding.

1 Introduction

NASA is currently investigating advanced life support systems for extended operation in future space habitats such as the space station or possible planetary sites. Since 1995, our group has been working at NASA's Johnson Space Center to provide intelligent control for advanced life support systems [3, 5]. These intelligent control systems have been realized by software agents using an architecture known as 3T [2] and were designed to run autonomously for months at a time. 3T is a layered control architecture whose top tier is a hierarchical task net (HTN) planner, the plans of which are executed through a reactive middle tier that in turn manages the sensors and actuators of the hardware via a low-level control tier. One such life support system is the advanced Water Recovery System (WRS). The WRS removes the organic and inorganic materials from waste water (hand wash, shower, urine and respiration condensate) to produce potable water.

In a previous paper, [1], we have explored safety-related issues for the design of the autonomous control software for the WRS system. These design issues include using adjustable autonomy to allow humans to interact with the agent safely, counteracting the slow degradation of hardware over time, being able to "safe" subsystems (put them into a shutdown or standby mode) in the event of power or communication failures, using checkpoints to quickly restore the WRS to nominal operations, and developing tools to help the human understand problem situations in order to recover from the anomaly.

In this paper, we focus on achieving the safe operation of the WRS by supporting the coordination of multiple humans and the control system, who may each take actions on the same underlying WRS system. Although the 3T-based automated control system operates the WRS hardware unattended most of the time, there are several cases in which humans must also take actions on the life support system. Actions that humans take can be either *manual* or *mediated*. *Manual* actions are those that the human carries out directly on the life support system hardware, for example, physically turning a valve. A human conducts *mediated* actions by giving instructions to the automation software, which carries out the actions. In contrast, *automated* actions are those taken by the control software during its normal operation without any requests from an external source. Manual and mediated actions are needed for two possible reasons (1) the action must be manual because the automation has no appropriate actuator or (2) the action could be carried out either by a human or via the software but is motivated by circumstances outside the scope of normal operation for the automation.

Challenges arise in coordinating humans and the control agent in their actions on the system because simultaneous or interleaved actions may be required or desired. The motivation for different agents to take different actions may arise from independent triggers or goals, and these actions may conflict with or impede each other. Further, it is difficult for humans to determine what actions other humans may be taking on the system because users may be located remotely from the WRS when taking mediated actions. It is also difficult for the autonomous control agent to determine what human agents are doing, both due to limited instrumentation of manual control inputs and due to the lack of models for manual actions. Such models might allow the control agent to map observed human actions to known WRS procedures for the purpose of predicting the human's next steps and maintaining safe operation of the WRS throughout the procedure.

In this paper, we present command and authorization services as implemented in a user support system for interacting with automated control agents called the Distributed Collaboration and Interaction (DCI) Environment. The goals of these services in DCI are (1) to decrease the risk of conflicting commands to the underlying physical system (2) to decrease the risk of interfering with the work of another agent (human or the control agent) pertaining to the underlying physical system, and (3) to decrease the risk of the system being put into a bad state by the action of any agent (for example, a state where pumps may be damaged by attempting to pull water from a blocked source). In order to achieve these goals, we offer software that assists a human user in performing mediated commands and in reconfiguring the system so that it is safe to perform commands (mediated or manual). This reconfiguration support includes adjusting the autonomy of the autonomous control system when necessary. The DCI environment also provides command lock-outs for possibly conflicting commands from different human users by selectively granting authorization to act on the system.

The following two sections provide an overview of the WRS system and the commands on this system currently supported by the DCI prototype. The paper then discusses how DCI supports command and authorization capabilities including detailed discussions of the authorization model, the need for authorization overrides, and support for reconfiguring the WRS and its control agent to accommodate human activities.

2 Water Recovery System (WRS) Overview

The WRS is composed of four subsystems shown in **Fig. 1**. These subsystems are loosely coupled, and their primary interdependencies are related to input and output of the water to be processed.

(1) The *biological water processor (BWP)* removes organic compounds and ammonia by circulating the water through a two-stage bioreactor. The first stage uses microbes to consume the organic material using oxygen from nitrate molecules. The second stage uses microbes to convert the ammonium to nitrate.

(2) The *reverse-osmosis (RO)* subsystem removes inorganic compounds from the output of the BWP, by forcing the water to flow at high pressure through a molecular sieve. The sieve rejects the inorganic compounds, concentrating them into brine. At the output of the RO, 85% of the water is ready for post-processing, and 15% of the water is brine.

(3) The air *evaporation system (AES)* removes the concentrated salts from the brine by depositing it on a wick, blowing heated air through the wick, and then cooling the air. The inorganic wastes are left on the wick and the condensate water is ready for post processing.

(4) The *post-processing system (PPS)* makes the water potable by removing the trace inorganic wastes and ammonium using a series of ion exchange beds and by removing the trace organic carbons using a series of ultra-violet lamps.

In total, the automated control system for the WRS manages more than 200 sensors (measuring pressure, temperature, air and water flow rates, pH, humidity, dissolved oxygen, and conductivity) and actuators (including pumps, valves, ultra-violet lamps, and heaters).

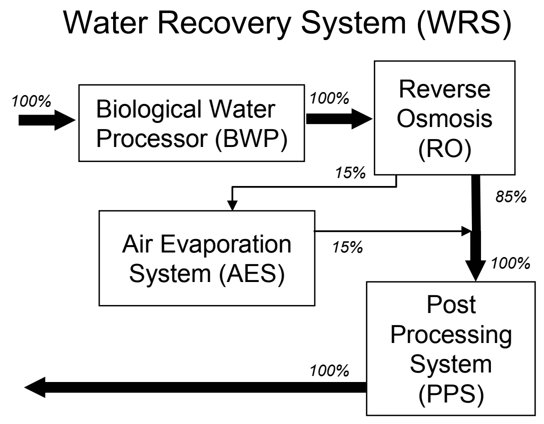
Water Recovery System (WRS)
100%
Biological Water Processor (BWP)
100%
Reverse Osmosis (RO)
15%
85%
Air Evaporation System (AES)
15%
100%
Post Processing System (PPS)
100%

Fig. 1. Schematic diagram of WRS system and subsystems

3 WRS Activities Supported

Our current work concerning command and authorization addresses the coordination of multiple humans with each other and with the automation before, during, and after the execution of human-initiated actions on the WRS hardware. We currently support these four human-initiated activities:

• *BWP nitrifier slough* – The biofilm that grows on the insides of the tubes in the nitrifying portion of the BWP will thicken over time, slowly constricting the passage of water and air. To minimize clogs, the control system periodically sloughs the biofilm by sharply increasing the airflow. This automatic slough is only partially effective, and eventually a human is required to manually slough the nitrifier using high pressure water flow. The configuration for this activity requires ensuring that water is flowing in the BWP as well as suspending the automatic shutdowns (ASDs) that the control automation will normally enact if tube pressure readings go outside the nominal range. The manual slough takes from twenty minutes to an hour to complete. The BWP nitrifier slough is a manual activity.
• *RO slough* – Inorganic deposits may accumulate inside the RO's tubular membranes. If the water flow is reversed, a small ball in each tube will slide along the tube length, sloughing this buildup away. The automated control system carries out this RO slough at a predetermined frequency. If the RO output quality degrades, a human may manually command the control system to slough the membranes again. Reconfiguration for this activity requires the RO to be shutdown. The RO slough takes four minutes to complete followed by a thirty minute purge of the RO subsystem. The RO slough is a mediated activity. This is the only mediated action the command and authorization service currently supports.
• *RO membrane change out* – Eventually the RO membranes lose their efficiency and must be physically replaced. The RO is shutdown, and the upstream and downstream subsystems are placed in standby mode. The change out takes approximately twelve hours to complete. The RO membrane change out is a manual activity.
• *BWP pressure calibration* – Pressure sensors are the primary input used to control the BWP. These sensors require calibration about every three months. In order to conduct the calibration, the BWP must be disconnected from the downstream subsystems and placed in a standby mode. The calibration procedure usually takes from four to six hours to complete. The BWP pressure calibration is a manual activity.

4 Commanding the WRS

When a human wishes to perform actions on the WRS using the command and authorization capability in the DCI environment, he or she requests the appropriate commanding permission for a particular activity. Throughout the paper, the term *authorization* implies a license to take action on the WRS. We use the term *commanding* to convey this authorization plus the concept of whether the system is ready for the execution of a particular activity associated with a pre-defined procedure. To grant commanding for a given activity, DCI must first, if possible, grant authorization for the set of manual or mediated actions (including reconfiguration actions) required by the activity, and then reconfigure the WRS hardware and control automation to the proper state required for the activity.

In the DCI environment, each user is represented by an Ariel agent [4], which acts as a liaison between the user and the rest of the software environment. An Ariel agent provides a human-centric interface into the software environment and provides a number of services including notification, task tracking, and location tracking. In particular, the Ariel agent provides a Command and Authorization Service, which assists its user with command and authorization requests. **Fig. 2** shows two Ariel agents, the WRS system, and components discussed in the upcoming subsections: the Command and Authorization Manager (CAM) and the Augmentation for Commanding (AFC).

4.1 Command and Authorization Manager (CAM)

The CAM accepts requests for commanding from users through their Ariel agents. Each request is associated with an activity that the user wishes to perform. The CAM first queries the AFC (see next subsection) for information about the effects of the requested activity as well as any configuration conflicts between the current system configuration and the configuration required for the activity. Section 5, below, describes how the CAM uses the results of this query to grant or deny authorization. If authorization is denied, this result is returned to the user along with a description of the configuration conflicts. If authorization is granted, and the user wishes to continue, the CAM asks the AFC to carry out any required reconfiguration on the WRS including orchestrating required manual actions. Once the reconfiguration, if any, is complete, the CAM informs the user through his or her Ariel that the WRS is ready to command. The user can then proceed with the actions required by the procedure for the requested activity. When the user has completed the activity, he or she requests the CAM to release commanding for the activity. The CAM informs the AFC that the

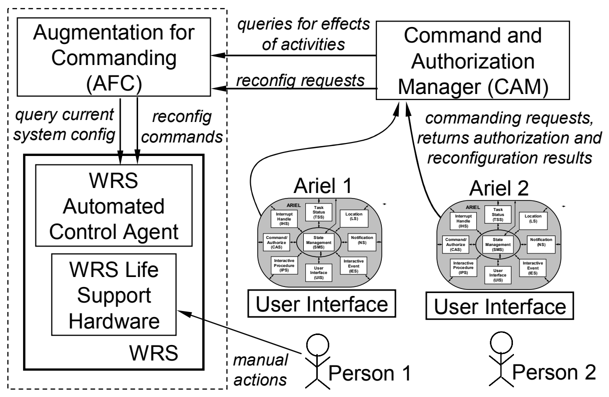
Augmentation for Commanding (AFC)
queries for effects of activities
reconfig requests
Command and Authorization Manager (CAM)
query current system config
reconfig commands
commanding requests, returns authorization and reconfiguration results
WRS Automated Control Agent
WRS Life Support Hardware
WRS
Ariel 1
Ariel 2
User Interface
User Interface
manual actions
Person 1
Person 2

Fig. 2. Implementation of Command and Authorization service in the DCI environment

activity's configuration is no longer required (which may result in additional reconfiguration of the WRS by the AFC to “un-configure” for the activity) and then releases the authorization.

4.2 Augmentation for Commanding (AFC)

The AFC is a piece of augmenting software in the DCI environment (shown by the dotted lines indicating coupling to the WRS). *Augmenting software* is tightly coupled to the automation through shared models or data but has its own processing resources. In this case, the AFC shares static models of both the physical WRS system and the procedures that can be performed on the system (including reconfiguration procedures). Using these models, the AFC can predict how various activities will affect the WRS. The AFC can also query the WRS control agent dynamically to get the current system configuration.

When the CAM queries the AFC about the effects of an activity, the AFC provides two results. First, the AFC decomposes the associated reconfiguration procedure (as well as the activity's procedure model, if available) to determine and return all components of the WRS that may be affected by the activity. In the current implementation, this result is highly abstracted and consists of an indicator for the highest-level system or subsystem that is affected. This system/subsystem approach is made extensible by also returning the specific decomposition of subsystems that are affected by the reconfiguration (in the future, subcomponents of the subsystems may also be used here). Second, the AFC queries the WRS automated control agent for the current system configuration (i.e., the current state of the eight valves and ten pumps in the WRS) and returns a list of conflicts between the current state and the state that would result from reconfiguration. The CAM uses the first result to determine whether to grant authorization for the activity, and passes the second set of results back to the user.

If the CAM asks the AFC to reconfigure the WRS for a requested activity, the AFC triggers the WRS control agent to perform the reconfiguration, if any. During the course of the reconfiguration, some manual actions may also be required. When it is time for a manual reconfiguration action, the WRS control agent, through the AFC, CAM, and the Ariel agent's user interface, requests the user to perform the action and waits for a return indication from the user that it is accomplished. This feedback from the user is needed because manually operated physical devices are not normally instrumented for computers, so manual actions are not easily observable by the software for tracking a user's progress in the reconfiguration. Once all reconfiguration actions have been completed, the CAM informs the user that the WRS is ready for commanding.

5 Managing Authorizations and Overrides

Authorization to act on the WRS is managed by the CAM. The CAM is centralized to provide synchronized access from multiple entities (various Ariel agents and, in the future, the automated control system itself) to a single model describing which entities hold which authorizations. In general, granting authorization to one entity for a given scope of action blocks other entities from receiving authorization overlapping that scope until the first authorization is released. This blocking authorization paradigm is a well-known technique and is applied here to prevent multiple entities from acting on the WRS simultaneously for activities within the same scope, which may therefore interfere with one another.

When possible, the CAM should authorize concurrent activities that can be achieved safely together. In our life support domain, crew time is a very valuable resource, and crew health is a top priority. Therefore we want to minimize the circumstances under which our system might unnecessarily block a crew member from performing an activity on the life support system or unnecessarily slow down that crew member. Further, our design philosophy must account for the nature and culture of space exploration in which crew safety is considered to be the top mission priority, above vehicle health and mission success. Since life support systems are required for crew safety, inadvertently taking actions that impede or interfere with crew life support can have a negative effect on crew safety. *Absolutely* preventing a crew member from performing *any* activity on a life support system could potentially be fatal, given an unforeseen circumstance or an emergency situation.

Therefore, our authorization design goal is to enhance safe operation of the life support system by helping to coordinate humans and the control agent to prevent unknowing or accidental conflicts. However, we are fully cognizant that a well trained and fully informed crew member should be allowed to override any blocking authorization that may exist, and take action, risking a conflict in order to achieve a possibly higher purpose. Consequently, our design has two components (1) determine which activities can be safely authorized for concurrent execution and allow the maximum concurrency possible, and (2) if an activity cannot be authorized because it cannot be guaranteed for safe execution in conjunction with other currently authorized activities, provide as much information as possible about potential conflicts to the user and allow the user to override the authorization. The following subsections discuss each of these design components in turn.

5.1 Authorizations

We believe that the maximum concurrency without risking conflicts can be achieved by authorizing activities Act1 and Act2 concurrently as long as (1) their configurations do not conflict (states of the hardware and software) and (2) no action taken for Act1 (during reconfiguration or the procedure itself) affects the same component or state value (i.e., valve position) as any action taken for Act2, and vice versa. For our initial approach, we used models already within the WRS control agent to support command and authorization and limited our development of new models. Unfortunately, (1) the existing models for the required configurations are not detailed enough to guarantee no conflicts (e.g., they have not been extended to include required operating characteristics of the automation) and (2) we do not have models of the procedures for activities that require only manual action.

Until we extend the activity models and reconfiguration models to overcome these limitations, we have initially adopted a conservative approach to authorization that works well with the existing models but does not allow the maximum possible authorization concurrency. The approach is conservative in that it locks authorization for an entire subsystem (e.g. the RO) if any component of that subsystem is affected by an activity (by the reconfiguration, or the activity itself if a model exists), and it locks authorization for the entire WRS if multiple subsystems or the dependencies between subsystems (e.g. water flow) are affected. For the small set of actions and scenarios we have considered thus far, the conservative nature of this approach has not been a disadvantage.

When a user requests commanding permission for a given activity from the CAM, the CAM obtains information from the AFC about the highest-level system or subsystem affected by the activity. The CAM translates the system/subsystem decomposition into a model of scopes for granted authorization. Let Φ be the set of all system components such that authorization can be assigned for the scope of that component. For the current implementation $\Phi = \{\text{WRS, BWP, RO, AES, PPS}\}$. For the variables x and y, let $x, y \in \Phi$. Let $Sub(x, y)$ define a predicate that indicates whether component x is a subsystem or subcomponent of component y in a hierarchical decomposition of the system. For the current implementation, the following hold: $Sub(\text{BWP, WRS})$, $Sub(\text{RO, WRS})$, $Sub(\text{AES, WRS})$, $Sub(\text{PPS, WRS})$.

Let α be the set of all agents (including humans and the automated control agent) that can act on the system. For the variables a and b, let $a, b \in \alpha$. Let $Auth(a, x)$ define a predicate indicating that agent a has authorization to act over the scope of system component x.

The CAM uses the following rule to assign authorizations: When b requests $Auth(b, x)$, then grant $Auth(b, x)$ if and only if no other agent holds the authorization for x, for any of x's subsystems, or for any component that has x as a subsystem. In other words, when $request(\,Auth(b, x)\,)$,

```
if $\forall a, \neg \mathrm{Auth}(a, x)$
        $\wedge \forall a, y, Sub(x, y) \Rightarrow \neg Auth(a, y)$
        $\wedge \forall a, y, Sub(y, x) \Rightarrow \neg Auth(a, y)$
then $Auth(b, x)$.
```

5.2 Overrides

If the CAM denies a user authorization to act on the system, the user should (by policy) wait until the authorization can be granted before taking any action. However, enforcing such a lockout could prevent a user from taking needed action in an emergency, which is a particularly troubling prospect with respect to a critical life support system. The development and use of more sophisticated models for the effects of activities on the system will allow us to avoid being overly conservative, maximizing the number of activities we can authorize concurrently. However, these advances will not address situations in which a low-priority ongoing activity may block authorization for an emergent higher-priority activity. We are currently working on building a user override capability for denied authorizations. The override capability should allow the user to obtain the authorization and perform the activity with *no less* protection from conflicts with newly arising tasks than the protection provided to a user granted a normal authorization. However, granting an override authorization is more complex than simply granting a new authorization that conflicts with existing authorizations. In particular, the specific areas of conflict must be identified and the appropriate users who currently hold authorizations must be notified about any potential problems that might arise in the context of the new override authorization. Determining the correct reconfiguration actions to take for an override situation also raises new questions. If configurations required for two simultaneously authorized activities conflict (i.e., require different state values or software modes), how should priority for setting these states be determined? We are currently working on a design to address these override issues. Explicit override capabilities are not currently supported in the prototype implementation.

The current implementation does allow overrides to occur, however, because the current WRS implementation offers limited options for enforcement of either denied authorizations or denied system access in general. There is some password protection for mediated actions, but anyone could theoretically walk up to the system at any time and, for example, power down a pump. We hope to improve enforcement as the override software support is developed. Suri et al describes relevant previous work on policy enforcement [7]. In the interim, when an authorization is denied, the CAM reports back to the requesting user the set of pre-existing authorizations that conflict with the request as well as the list of conflicts between the current system configuration and the requested activity's configuration. The highly trained user can consider this information to determine how to proceed. He or she may ask other users holding a conflicting authorization to release it, or he or she may proceed *manually* with the desired reconfiguration and activity with foreknowledge of possible conflicts that may arise. Although much work remains, making users aware of possible conflicts arising from ongoing activities by other users on the WRS is an important first step toward supporting the coordination of multiple humans and an automated control agent working on the same underlying physical system.

5.3 A Note on Security

The current CAM implementation assumes that every entity requesting authorizations possesses the necessary credentials (authentication, skills, and/or certificates) for the authorization to be granted. We would like to add credential checking in the future. However, it is not currently critical in our application because (1) we assume all possible users (NASA crew) are highly trained and (2) our authorization process is used primarily for coordination rather than access control enforcement. Although users must log in to use the DCI environment (authentication), they can currently act on the WRS by circumventing DCI completely. Users are motivated to request commanding permission through DCI primarily to minimize the risk of conflicts for themselves and the control agent and to obtain assistance from the AFC in reconfiguring the WRS hardware and the control agent for the desired activity. However, the users currently do not *need* the system's permission to take action.

6 Reconfiguration for Commanding

Reconfiguration for commanding is managed by the AFC. The AFC is coupled to the WRS automated control agent and shares its static models of both the physical WRS system and the procedures that can be performed on the system (including reconfiguration procedures). Using these models, the AFC can predict how various activities will affect the WRS. The AFC can also query the WRS control agent dynamically to get the current system configuration and it can trigger the WRS control agent to take actions to carry out any reconfiguration necessary to prepare for an activity. In general, the reconfiguration process may include setting the states of particular hardware such as valves open/closed or pumps on/off, adjusting the autonomy of the automation to allow for manual actions [6], bringing the state of the system to a particular point such as getting tube pressures or heater temperatures within a specified range, or commanding a subsystem to a particular processing mode. The current implementation handles a subset of these types of reconfiguration actions and affects both hardware (the states of eight valves and ten pumps) and software (the operating characteristics of the automated control system). Actions required to achieve the reconfiguration necessary for each of these activities may be either manual or mediated. Note that *mediated* actions are performed by the control agent, but triggered externally, and they can be initiated by a human or by external software. Actions taken by the WRS control agent in the course of reconfiguration are examples of mediated actions that are initiated by external software (the AFC).

We found that models of reconfiguration procedures could be used to (1) determine what parts of the WRS would be affected by (reconfiguring for) an activity and (2) allow the AFC to trigger the WRS control agent to perform the reconfiguration necessary. Except for mediated activities, such as the RO slough, in which the control agent performs the actions in the body of the activity itself, models of reconfiguration procedures were not originally developed for the WRS control agent because they were not necessary for autonomous operation. In support of the DCI commanding capability, we added models of the reconfiguration procedures for the other three activities described above in Section 3.

The AFC ensures that the WRS maintains the configuration, as a whole, required to support all of the currently authorized activities. If authorization were allowed for only one activity any given time, the AFC could support reconfiguration for this activity by first triggering the WRS to execute the reconfiguration procedure for that activity after authorization is granted and then triggering the WRS to execute the reconfiguration procedure to return to nominal operation before authorization is released. However, since multiple authorizations should be supported, the AFC must unify the configuration state required for all concurrent authorizations. The following paragraphs describe how the AFC and WRS control agent together achieve the desired unified configuration for all currently authorized activities.

Let C be the set of all components in the WRS system. In general, this set may include hardware (values and pumps), software modules, measurable operating characteristics (such as tube pressure), or abstractions of groups of system pieces such as subsystems. To apply the reasoning presented here, the members of C must be independent and separable. This means, for example, that no pump listed as a member of C can be a part of the BWP subsystem if the BWP subsystem is also a

C can be a part of the BWP subsystem if the BWP subsystem is also a member of *C*. For the variable *c*, let $c \in C$.

Let S be the set of all states that components in the WRS system can take. Examples of possible state values in S may include ON, OFF, OPEN, CLOSED, <180psi, STANDBY, etc. For the variable *s*, let $s \in S$.

Let the tuple (*c*, *s*) be a component-state pair[1] in which component *c* takes on the state value *s*. Let *R* be a set of *n* component-state pairs representing a configuration state in the WRS. Components in *C* that are not included in any element of *R* have no specific state requirement for that configuration (i.e., the states of these components are *don't cares* in the configuration).

$$R = \{(c^1, s^1), (c^2, s^2), \ldots (c^n, s^n)\}.$$

The desired configuration state for a given activity can be determined by examining the reconfiguration procedure for that activity. The AFC keeps a prioritized list of the configurations required. The lowest priority configuration (priority 0) is the normal operating configuration (nominal-ops) during which no activities are currently authorized. When any activity is authorized, its configuration is added to the list and given the next highest priority above nominal-ops (priority 1). Once support for override authorizations is implemented, the configuration for activities requiring overrides would be added to the list at even higher priority levels. Assigning these priorities correctly for configurations related to overrides is an open research issue. Given this prioritized list, the unified configuration is determined by stepping through each configuration, starting with the lowest priority configuration (configurations with the same priority may be processed in any order), and adding its component-state pairs to the unified result. As each component-state pair is added, it will overwrite any pair containing the same component in the unified configuration. Therefore, in the final unified configuration, only the highest priority state for each component will be included. The AFC triggers the WRS control agent to apply this desired unified configuration each time a change in authorizations occurs.

[1] For brevity in this discussion, we will not explicitly disallow unrealistic component-state pairs such as (TUBE1PRESSURE, ON) or (VALVE2, <180psi).

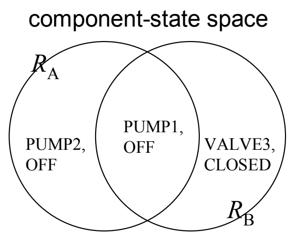

Fig. 3. Venn diagram of required configurations, R, for Activity A and Activity B

Consider the following example containing no overrides: Assume, that the nominal-ops configuration for normal operation during which no activities are authorized is R_N = {(PUMP1,ON),(PUMP2,ON),(VALVE3,OPEN)}. Activity A requires configuration R_A = {(PUMP1,OFF),(PUMP2,OFF)} and Activity B requires configuration R_B = {(VALVE3,CLOSED),(PUMP1,OFF)}. These configurations overlap (contain the same component holding the same state value) as shown in **Fig. 3**, but do not conflict (do not contain the same component with different state values).

Therefore, these activities A and B could be authorized concurrently[2] with the same configuration priority. **Table 1** shows some example sequences of authorizations for these activities (assuming concurrent authorization is supported) and how the configuration of the WRS would change to accommodate these authorizations.

Table 1. Possible authorization sequences with resulting configuration changes

Time	Authorized Activities	Prioritized Configurations	Unified Desired Configuration	Actions Taken
t_0	none	(R_N)	PUMP1, ON PUMP2, ON VALVE3, OPEN	none
t_1	A	(R_N, R_A)	PUMP1, OFF PUMP2, OFF VALVE3, OPEN	*turn off* PUMP1 *turn off* PUMP2
t_2	none	(R_N)	PUMP1, ON PUMP2, ON VALVE3, OPEN	*turn on* PUMP1 *turn on* PUMP2
t_3	B	(R_N, R_B)	PUMP1, OFF PUMP2, ON VALVE3, CLOSED	*turn off* PUMP1 *close* VALVE3
t_4	B, A	(R_N, R_B, R_A)	PUMP1, OFF PUMP2, OFF VALVE3, CLOSED	*turn off* PUMP2
t_5	A	(R_N, R_A)	PUMP1, OFF PUMP2, OFF VALVE3, OPEN	*open* VALVE3

[2] However, for our currently implemented conservative authorization model in which users block an entire subsystem if they affect a single component in that subsystem, these activities would not be authorized concurrently.

7 Conclusions

The command and authorization services in the DCI environment are designed to support the safe operation of advanced life support systems and their intelligent control agents by enhancing the coordination among multiple humans and these control agents. This work is still preliminary, but supports future evaluation with respect to safety metrics and guarantees. Our prototype system makes users aware of possible conflicts arising from ongoing activities by other users on the WRS system. We have developed and implemented a conservative policy for granting authorization to act on the system, which ensures that no more than one user at a time has authorization at a given scope. Further, as an integral part of processing a human's request to perform an activity on the physical system, we provide previously unavailable assistance in reconfiguring the system for that activity. By suspending or modifying automatic responses in the control system for the duration of human-initiated activities, we have also enhanced coordination between humans and the automation.

We plan to enhance our conservative authorization policy in the future as we develop improved models of the effects of human activity on the life support system and therefore better understand possible sources of conflict. We would like to further enhance our authorization capabilities by supporting credential checking as well as authorization enforcement and override capabilities. Finally, we plan to extend this work to better support coordination with the autonomous system. This additional support would include (1) extending supported activities to those containing a mixture of manual, mediated, and automated actions, (2) making more extensive use of the adjustable autonomy and traded control capabilities of the automation, (3) granting explicit authorizations to the automation in addition to humans such that humans are protected from unknowingly acting on the system when the automation is performing a critical operation, and (4) integrating command and authorization with control planning for the autonomous system and task planning for the human to avoid redundant reconfiguration. Although much work remains to fully support safe human commanding and authorization in coordination with autonomous systems, the preliminary work presented in this paper provides both enhanced capabilities and encouragement that we have defined a reasonable path forward.

8 Acknowledgements

This work was performed under the project titled "Distributed Crew Interaction with Advanced Life Support Control Systems" and supported by the Human-Centered Computing area of NASA's Intelligent Systems Program, which is under the direction of Dr Michael Shafto.

References

1. R. P. Bonasso, "Safe Agents for Life Support," in *Proc. Workshop on Safe Agents at AAMAS'03*, Melbourne, Australia, 2003, pp.

2. R. P. Bonasso, J. R. Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. G. Slack, "Experiences with an Architecture for Intelligent, Reactive Agents," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 237-256, 1997.

3. R. P. Bonasso, D. Kortenkamp, and C. Thronesbery, "Intelligent Control of A Water Recovery System: Three years in the Trenches," *AI Magazine*, vol. 24, 2002.

4. C. E. Martin, D. Schreckenghost, R. P. Bonasso, D. Kortenkamp, T. Milam, and C. Thronesbery, "An Environment for Distributed Collaboration Among Humans and Software Agents," in *Proc. 2nd International Conference on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, 2003, pp. 1062-1063.

5. D. Schreckenghost, D. Ryan, C. Thronesbery, R. P. Bonasso, and D. Poirot, "Intelligent Control of Life Support Systems for Space Habitats," in *Proc. Tenth Conference on Innovative Applications of Artificial Intelligence*, Madison, WI, 1998, pp. 1140-1145.

6. D. Schreckenghost, C. Thronesbery, R. P. Bonasso, D. Kortenkamp, and C. E. Martin, "Intelligent Control of Life Support for Space Missions," *IEEE Intelligent Systems*, vol. 17, pp. 24-31, 2002.

7. N. Suri, J. M. Bradshaw, M. Burstein, A. Uszok, B. Benyo, M. Breedy, M. Carvalho, D. Diller, P. Groth, R. Jeffers, M. Johnson, S. Kulkarni, and J. Lott, "DAML-based Policy Enforcement for Semantic Data Transformation and Filtering in Multi-agent Systems," in *Proc. Second International Joint Conference on Autonomous Agents and MultiAgent Systems*, Melbourne, Australia, 2003, pp. 1132-1133.